6.5 Machine de Turing universelle

On peut construire une M.T. *universelle* U qui,étant donné $< M, w >$ exécute M sur w : $U(< M, w >) = M(w)$. Le codage $< M, w >$ est défini par :

- $< M, w >=< M >; < w >$
- $< (Q, q_0, \Sigma, \delta, \{B, \$\}) >=< Q > d_\Sigma < \Sigma > d_\delta < \delta > f_\delta$
- $< Q >= \overbrace{0 \cdots 0}^{1+\lfloor \log_2 |Q| \rfloor}$; il y a autant de zéros que dans l'écriture en binaire du nombre d'éléments de Q. On suppose les états codés par des entiers $1, ..., |Q|$ et $q_0 = 1$, avec des 0 devant de façon à avoir tous même longueur
- $< \Sigma >= \overbrace{0 \cdots 0}^{\lfloor \log_2 |\Sigma| \rfloor}$ On suppose que B et \$ correspondent aux entiers 1 et 10 en binaire. Tous les symboles de Σ sont représentés par des entiers en binaire, complétés le cas échéant par des 0 à gauche, de manière à ce qu'ils aient tous même longueur.
- $< \delta >$ est codé comme suit : c'est une séquence de règles écrites sous la forme $c(q), c(a) \mapsto c(q'), c(b), m$; où le codage des états et symboles est par leur représentation binaire
- $< aw >= c(a) < w >$.

Le codage $< M, w >$ peut aussi n'utiliser qu'un alphabet de deux symboles : il suffit de recoder l'ensemble des symboles utilisés en binaire, ce que nous ne faisons pas pour des raisons de lisibilité.

Théorème 6.5.1 *Le langage* universel

$L_U = \{< M, w > \mid w \in L(M), M \text{ machine de Turing sur un alphabet } \Sigma, w \in \Sigma^*\}$

est récursivement énumérable.

Preuve:

On ne donne que les grandes lignes.

On utilise trois rubans sans perdre de généralité d'après le théorème 6.4.2.

On utilisera l'alphabet $\Sigma_U = \{0\ 1\ ;\ ,\ d_\Sigma\ d_\delta\ f_\delta\ \mapsto \to \downarrow \leftarrow\ B\ \$\}$.

Le codage $c(\gamma)$ d'une configuration de la machine $M = (Q, \Sigma, q_0, \delta, B, \$)$ est défini par $c(w, q, w') = \$ < w >, c(q), < w' >$. On définit M_U de manière à ce que, pour toute machine M, si γ est une configuration de M alors $\gamma \vdash_M \gamma'$ si et seulement si $(q_s, (\epsilon, \$ < M >), (\epsilon, c(\gamma)), (\epsilon, \$)) \vdash^*_{M_U} (q_s, (\epsilon, \$ < M >), (\epsilon, c(\gamma')), (\epsilon, \$))$ pour un certain état q_s de la machine M_U.

La machine universelle commence par écrire la configuration initiale de M sur le deuxième ruban : à partir de $(q_0^U, (\epsilon, < M, w >), (\epsilon, \$))$ on définit (ce que nous ne détaillons pas ici) des transitions permettant d'atteindre la configuration $(q_s, (\epsilon, < M >), (\epsilon, \$c(\gamma_0))$ où γ_0 est la configuration initiale de M sur w. Il s'agit essentiellement d'avancer la première tête de lecture jusqu'à la partie de la donnée correspondant à $< w >$, puis de déplacer $< w >$ sur le deuxième ruban. Notons aussi que copier q_0 s'effectue en se plaçant au début de $< Q >$ sur le premier ruban, recopiant $< Q >$, puis en changeant le dernier bit à 1.

Puis, si $\gamma \mapsto_M \gamma'$, on procède comme suit :

1. Se placer au début (du codage) de la première transition sur le premier ruban et au début (du codage) de l'état de M sur le second ruban.
2. On progresse simultanément sur les deux rubans tant que les symboles sont identiques. Comme f_δ apparaît à droite de la tête de lecture sur le premier ruban et pas sur le deuxième ruban, on atteint une situation dans laquelle les deux symboles ne sont pas identiques. On distingue alors suivant les cas :
   (a) si les symboles différents sur les deux rubans sont dans $\{0, 1\}$, alors on avance la tête de lecture sur le premier ruban jusqu'au début de la prochaine transition (repérée par ";") s'il y en a une. Sinon, la machine s'arrête (aucune transition n'est possible depuis la configuration de M codée sur le deuxième ruban). Sur le deuxième ruban, on revient à la position du début du codage de l'état de M (repéré par ",").
   (b) Sinon : on a trouvé une transition possible. Soit $(q_1, (\alpha c(q)", "c(a) \mapsto , c(q')", "c(b)\beta), (< w > ", ", c(q)", "c(a) < w' >))$ la configuration de M_U. Sur le premier ruban, on se déplace jusqu'au prochain symbole de déplacement m, que l'on mémorise dans l'état. Selon ce symbole,
   — Si $m = \downarrow$, on recopie $c(q')", "c(b)$ sur le deuxième ruban
   — Si $m = \to$ on recopie $c(b)$ sur le deuxième ruban (en écrasant $", "c(q)$) puis on recopie $", "c(q')", "$ sur le deuxième ruban. Comme tous les états et tous les symboles ont même longueur, on se retrouve avec la configuration $(< w > c(b)", "c(q')", ", < w' >)$

sur le deuxième ruban. Si $< w' >= \epsilon$ (i.e., le symbole pointé par la tête du deuxième ruban est un blanc), on copie $c(B)$ sur le deuxième ruban. Recopier $c(B)$ s'effectue comme la copie de $c(q_0)$: on se place au début de $< \Sigma >$ sur le premier ruban, on recopie $< \Sigma >$, puis on change le dernier bit à 1.

— Si $m =\leftarrow$, soit $w = w_1 a'$. On recule de $| < \Sigma > |+1$ symboles sur le deuxième ruban (ce qui est possible en utilisant $< \Sigma >$ sur le premier ruban), puis on recopie $| < \Sigma > |$ symboles sur le troisième ruban (autrement dit, on recopie $c(a')$ sur le troisième ruban), on revient à nouveau $| < \Sigma > |$ symboles en arrière sur le deuxième ruban, puis on recopie $",", "c(q')", "$ sur le deuxième ruban (ceci écrase, au moins en partie, $c(a')$, qui a auparavant été sauvegardé sur le troisième ruban) . Puis on recopie le troisième ruban sur le deuxième. Enfin, on recopie $c(b)$ sur le deuxième ruban. On arrive ainsi dans la configuration $(< w_1 > ", "c(q')", "c(a')c(b), < w' >)$ su second ruban.

3. On revient au début des deux rubans et on efface le troisième ruban.

On se retrouve alors dans le codage de la configuration suivante de M.

Remarquons que les codages sont calculables. Par exemple :

Lemme 6.5.1 *La fonction qui à $< M >$ associe $< M, < M >>$ est calculable.*

En effet, il suffit de recoder $< M >$ une deuxième fois en utilisant les codages des symboles 0 et 1.

Exercice 162
Expliciter le codage

Exercice 163
Montrer que $\{< M > \mid M$ Machine de Turing $\}$ est récursif.

Théorème 6.5.2 *L_u n'est pas co-récursivement énumérable (et donc pas récursif).*

$\overline{L_u} = \{< M, w > \mid w \notin L(M)\}$. Si ce langage était r.é, soit M_N une machine de Turing telle que $L(M_N) = \overline{L_u}$. On construit alors une machine de Turing D telle que $L(D) = \{< M > \mid < M >\notin L(M)\}$: sur la donnée $< M >$, D simplement recopie le code de M pour obtenir $< M, < M >>$ puis applique M_N. Mais alors,

$$< D >\in L(D) \Longleftrightarrow < D >\notin L(D)$$

Ce qui est absurde.

6.6 Problème de l'arrêt

Soit $L_{arret} = \{< M, x > \mid M(x) \neq \perp\}$.

Théorème 6.6.1 *L_{arret} est récursivement énumérable et pas co-récursivement énumérable (donc indécidable).*

Tout d'abord, L_{arret} est récursivement énumérable : considérons la machine M_{arret} qui, étant donné $< M, x >$, simule la machine universelle et, quand celle-ci est sur le point d'accepter ou de rejecter, accepte. M_{arret} accepte L_{arret}.

Si L_{arret} était co-récursivement énumérable, soit $\overline{M_{arret}}$ une machine qui accepte $\overline{L_{arret}}$. On construit alors la machine H qui accepte $\{< M > \mid M(< M >) = \perp\}$ comme suit : H commence par dupliquer $< M >$, écrivant $< M, < M >>$, puis applique $\overline{M_{arret}}$. Si $\overline{M_{arret}}$ est sur le point de s'arrêter en **reject**, H entre dans un état spécial, où H se déplace indéfiniment vers la droite sans rien faire (et donc ne s'arrête pas).

$< H > \in L(H)$ si et seulement si $H(< H >) = \perp$. Mais, comme H ne rejette aucun mot, $H(x) = \perp$ si et seulement si $x \notin L(H)$. Donc $H(< H >) = \perp$ si et seulement si $< H > \notin L(H)$. Absurde. Il n'existe donc aucune machine de Turing qui accepte $\overline{L_{arret}}$.

6.7 Réductions

D'une manière générale, on posera les problèmes de la façon suivante :

Donnée : $D \in S$

Question : Q

où S est un ensemble récursif et $Q \subseteq S \subseteq \Sigma^*$. Il s'agit d'une façon d'écrire l'ensemble $\{D \in S \mid Q\}$ où Q est cette fois vu comme un prédicat. On se passe le plus souvent de parler du codage de la donnée. Par exemple, on écrit

Donnée : M, w où M est une machine de Turing et $w \in \{0,1\}^*$

Question : M s'arrête-t-elle sur w ?

est indécidable.

Cet énoncé est une façon de dire que $\{< M, w > \mid M(w) \neq \perp\}$ n'est pas récursif.

Pour montrer qu'un problème, donné sous cette forme, est indécidable, on procède souvent par *réduction* d'un problème connu pour être indécidable : Si P_1 est le problème

Donnée : $D_1 \in S_1$

Question : Q_1

est indécidable, et P_2 est le problème

Donnée : $D_2 \in S_2$

Question : Q_2

Pour prouver que P_2 est indécidable, il suffit d'exhiber une fonction récursive $f : S_1 \to S_2$ telle que, pour toute donnée $D_1 \in S_1$, $D_1 \in Q_1 \iff f(D_1) \in Q_2$. Voici un premier exemple

Proposition 6.7.1 *Le problème suivant est indécidable :*

L'arrêt universel :

Donnée *: une machine de Turing M*

Question *: M s'arrête-t-elle sur toutes les données ?*

On réduit le problème de l'arrêt : on considère la fonction f qui à $< M, x >$ associe $< M_x >$ où M_x est la machine qui écrit x puis simule M sur x (sans tenir compte de l'entrée). f est une fonction calculable : il suffit en effet d'ajouter$|x| + 2$ états à M, qui permettent d'effacer le ruban et d'écrire x. De plus, M_x s'arrête sur toute entrée si et seulement si M s'arrête sur x.

6.7.1 Extension de la notion de décidabilité

La notion de réduction ne requiert pas a priori que l'ensemble S dans lequel varie la donnée du problème soit récursif. Cependant, pour que le probème de décision soit équivalent à la décision de $Q \cap S$, il faut que S soit récursif.

Definition 6.7.1 *Si $S, Q \subseteq \Sigma^*$, le problème*

Donnée : $D \in S$

Question : Q

est indécidable, s'il n'existe aucune machine de Turing M qui s'arrête sur toute donnée $d \in S$ et telle que $L(M) \cap S = Q \cap S$.

Lorsque S est récursif, un tel problème est décidable, si et seulement si $Q \cap S$ est récursif.

Les méthodes de réduction permettent de prouver l'indécidabilité dans ce sens plus général.

6.8 Théorème de Rice

Théorème 6.8.1 *Si $\mathcal{P}$ est une propriété non triviale des langages récursivement énumérables, alors $\mathcal{P}$ est indécidables.*

Nous donnons ci-dessus l'énoncé "classique", mais en voici une version plus précis :

Pour tout ensemble $\mathcal{P}$ de (codes de) machines de Turing tel que :

1. pour toutes machines M, M' si $< M > \in \mathcal{P}$ et $L(M) = L(M')$ alors $< M' > \in \mathcal{P}$
2. on peut exhiber des machines M_1, M_2 telles que $< M_1 > \in \mathcal{P}$ et $< M_2 > \notin \mathcal{P}$

alors le problème suivant est indécidable :

Donnée : $< M >$

Question : $< M > \in \mathcal{P}$?

Pour prouver ce résultat, nous utiliserons une réduction du problème de l'arrêt, un type de réduction qui sera très fréquemment utilisé par la suite.

Soit $\mathcal{P}$ un sous-ensemble non-trivial des langages récursivement énumérables. $\mathcal{P}$ est récursif ssi $\overline{\mathcal{P}}$ est récursif. On peut donc supposer sans perte de généralité que $\emptyset \notin \mathcal{P}$. Comme $\mathcal{P}$ est non vide, soit $L \in \mathcal{P}$ et M_L une machine qui accepte L.

On réduit le problème de l'arrêt

Donnée : $< M, x >$ où M est une machine de Turing sur Σ et $x \in \Sigma^*$

Question : M s'arrête-t-elle sur x ?

au problème de l'appartenance à $\mathcal{P}$ en construisant, à partir de la donnée $< M, x >$ du problème de l'arrêt, une donnée M_x du problème de l'appartenance à $\mathcal{P}$, de telle manière à ce que $M_x \in \mathcal{P}$ ssi M s'arrête sur x.

Soit M_{arret} une machine qui accepte L_{arret}. M_x est une machine qui, sur la donnée y, commence par simuler M sur x et, en cas d'arrêt, simule ensuite M_L sur y. On remarque que $L(M_x) \in \{L, \emptyset\}$ et donc $L(M_x) \in \mathcal{P}$ ssi M s'arrête sur x.

Exercice 164 (4)
Dire si les problèmes suivants sont décidables ou non (justifier) :

1. **Donnée :** le code $< M >$ d'une machine de Turing
   Question : M s'arrête-t-elle sur le mot vide ?
2. **Donnée :** le code $< M >$ d'une machine de Turing
   Question : M s'arrête-t-elle sur au moins une donnée ?
3. **Donnée :** les codes $< M >$ et $< M' >$ de deux machines de Turing
   Question : $L(M) = L(M')$?
4. **Donnée :** le code $< M >$ d'une machine de Turing et un mot w.
   Question : Est ce que la machine M boucle sur w ?
   On dit que M boucle sur w si, de la configuration initiale γ_0 on peut atteindre une configuration γ, de laquelle on peut à nouveau atteindre γ en au moins une étape : $\gamma_0 \vdash_M^* \gamma \vdash_M^+ \gamma$.
5. **Donnée :** le code $< M, w >$ d'une machine de Turing et d'un mot w et un entier n (en base 2)
   Question : M accepte-t-elle w après au plus n transitions ?
6. **Donnée :** le code $< M, w >$ d'une machine de Turing et d'un mot w et un entier n (en base 2)
   Question : M accepte-t-elle w après au moins n transitions ?
7. **Donnée :** le code $< M >$ d'une machine de Turing
   Question : Est ce que le complémentaire de $L(M)$ est récursivement énumérable ?
8. **Donnée :** Le code d'une machine de Turing M_1 et le code d'une machine de Turing M_2 qui s'arrête, pour tout mot d'entrée w après au plus $2 \times |w|$ transitions
   Question : Pour tout mot w, $M_1(w) = M_2(w)$
9. **Donnée :** Le code d'une machine de Turing M
   Question : il existe deux mots w_1, w_2 de même longueur tels que $w_1, w_2 \in L(M)$.
10. **Donnée :** Le code d'une machine de Turing M
    Question : M calcule-t-elle en temps polynomial ?

Exercice 165 (5)
Dire, en le justifiant, si les problèmes suivants sont décidables :

1. **Donnée :** Le code d'une machine de Turing M qui s'arrête sur toutes ses entrées.
   Question : M calcule en temps polynômial

2. **Donnée :** Le code de deux machines de Turing M_1, M_2 qui calculent en temps polynomial.

   Question : $L(M_1) \cap L(M_2) = \emptyset$

Exercice 166 (5)
Un *automate linéairement borné* est une machine de Turing qui, lorsqu'elle lit un blanc, écrit un blanc et se déplace vers la gauche.

Montrer que l'arrêt universel des automates linéairement bornés est indécidable :

Donnée : $< M >$ où M est un automate linéairement borné

Question : est ce que M s'arrête sur toute donnée ?

Exercice 167 (5)
Donner une fonction calculable dont l'image est indécidable.

Exercice 168
Un *système de réécriture (de mots)* $\mathcal{R}$, sur l'alphabet Σ est un ensemble fini de paires de mots $(u_i, v_i) \in \Sigma^* \times \Sigma^*$. La *relation de réécriture* associée à un tel système de réécriture est la relation $\rightarrow_{\mathcal{R}} \subseteq \Sigma^* \times \Sigma^*$ définie par

$$u \rightarrow_{\mathcal{R}} v \quad \text{ssi} \quad \exists i, \exists w, x \in \Sigma^*,\ u = wu_ix \wedge v = wv_ix$$

La *relation de réduction* $\xrightarrow[\mathcal{R}]{*}$ associée à un système de réécriture $\mathcal{R}$ est la fermeture réflexive et transitive de la relation de réécriture :

$$\begin{array}{lll} \xrightarrow[\mathcal{R}]{*} & = & \bigcup_{n=0}^{+\infty} \xrightarrow[\mathcal{R}]{n} \\ u \xrightarrow[\mathcal{R}]{0} v & \text{ssi} & u = v \\ u \xrightarrow[\mathcal{R}]{n} v & \text{ssi} & \exists w. u \rightarrow_{\mathcal{R}} w \wedge w \xrightarrow[\mathcal{R}]{n-1} v \end{array}$$

Par exemple, soit $\mathcal{R} = \{c \rightarrow abaca; c \rightarrow bcbabab; aca \rightarrow d;\ bcb \rightarrow d; ada \rightarrow d; bdb \rightarrow b\}$,sur l'alphabet $\Sigma = \{a, b, c, d\}$. $c \xrightarrow[\mathcal{R}]{*} d$, en effet :

$$c \rightarrow_{\mathcal{R}} abaca \rightarrow_{\mathcal{R}} ababcbababa \rightarrow_{\mathcal{R}} abababacababababa \rightarrow_{\mathcal{R}} abababdbababa \rightarrow_{\mathcal{R}} abababadababa \rightarrow_{\mathcal{R}} ababdbaba \rightarrow_{\mathcal{R}} abadaba \rightarrow_{\mathcal{R}} abdba \rightarrow_{\mathcal{R}} ada \rightarrow_{\mathcal{R}} d$$

Montrer que le problème suivant, appelé *accessibilité* est indécidable :

Donnée : deux mots $u, v \in \Sigma^*$, un système de réécriture $\mathcal{R}$

Question : est-ce que $u \xrightarrow[\mathcal{R}]{*} v$?

Exercice 169

1. Si $n \in \mathbb{N}$, on note $\overline{n}$ le mot 0^n, de longueur n.
   Soit M une machine de Turing. Montrer qu'on peut construire une machine M_v telle que :
   (a) étant donné $w = \gamma_1 \# \overline{n} \#$, M_v vérifie que γ_1 est une configuration de M accessible en n étapes par M à partir de la configuration initiale $(\epsilon, q_0^M, \$_M)$ et s'arrête dans la configuration $(\$_v, q_1, \gamma_2 \# \overline{n+1} \#)$ si c'est le cas, où $\gamma_1 \vdash_M \gamma_2$

(b) quelle que soit la configuration γ de M_v (même une configuration inaccessible), il n'existe aucune suite infinie $\gamma_i, i \in \mathbb{N}$ telle que $\gamma_0 = \gamma$ et, pour tout i, $\gamma_i \vdash_{M_v} \gamma_{i+1}$.

2. Montrer que le problème suivant est indécidable :

Donnée : Une machine de Turing M

Question : Existe-t-il une suite infinie de configurations (w_i, q_i, x_i) telle que, pour tout i, $(w_i, q_i, x_i) \vdash_M (w_{i+1}, q_{i+1}, x_{i+1})$?

Exercice 170 (5)
Un *Système de Post* P est donné par un ensemble fini de paires de mots $(\alpha, \beta) \in (\Sigma^*)^2$ et un *mot de départ* γ. Une *configuration* d'un tel système est un mot $w \in \Sigma^*$. On peut effectuer un mouvement de w vers w' s'il existe une paire (α, β) dans le système et un mot δ tels que $w = \alpha\delta$ et $w' = \delta\beta$.

Montrer que le problème

Donnée : un système de post P et un mot w

Question : peut-on atteindre une configuration w de P (après un nombre quelconque de mouvements)

est indécidable.

Exercice 171 (7)
Soit $\mathcal{P}$ une propriété des langages récursivement énumérables. ($\mathcal{P}$ est un ensemble de codes de machines de Turing $< M >$ tel que, si $L(M) = L(M')$ et $< M > \in \mathcal{P}$, alors $< M' > \in \mathcal{P}$. Par abus de langage on dira alors que $L \in \mathcal{P}$ s'il existe une machine de Turing M telle que $L(M) = L$ et $< M > \in \mathcal{P}$)

1. Montrer que, s'il existe deux langages récursivement énumérables $L_1 \subseteq L_2$ tels que $L_1 \in \mathcal{P}$ et $L_2 \notin \mathcal{P}$, alors $\mathcal{P}$ n'est pas récursivement énumérable.
2. Montrer que s'il existe $L \in \mathcal{P}$ tel que, pour tout $L' \subseteq L$, ou bien L' est infini, ou bien $L' \notin \mathcal{P}$, alors $\mathcal{P}$ n'est pas récursivement énérable.
3. On considère une application c (calculable) injective qui associe à chaque sous-ensemble fini de Σ^* un mot de Σ^*. Montrer que, si $\mathcal{P}$ est récursivement énumérable, $\mathsf{Fin}(\mathcal{P}) = \{c(L)(M) | < M > \in \mathcal{P}, L(M) \text{ fini } \}$ est récursivement énumérable.
4. En déduire que $\mathcal{P}$ est récursivement énumérable si et seulement si les trois propriétés suivantes sont satisfaites :
   (a) $\mathsf{Fin}(\mathcal{P})$ est récursivement énumérable
   (b) Tout langage de $\mathcal{P}$ contient un langage fini dans $\mathcal{P}$
   (c) Pour tous langages $L \subseteq L'$ récursivement énumérables, si $L \in \mathcal{P}$, alors $L' \in \mathcal{P}$.
5. Montrer que les propriétés suivantes des langages récursivement énumérables ne sont pas récursivement énumérables :
   (a) $\mathcal{P} = \{\emptyset\}$
   (b) $\mathcal{P} = \Sigma^*$
   (c) $\mathcal{P}$ est l'ensemble des langages finis

(d) $\mathcal{P}$ est l'ensemble des langages reconnaissables.

(e) $\mathcal{P}$ est l'ensemble des langages récursifs.

(f) $\mathcal{P}$ est l'ensemble des langages récursivement énumérables, mais pas récursifs

Exercice 172 (5)
Une *machine à registres* est donnée par un ensemble fini d'états (contenant les états **accept** et **reject**), un état initial, N *registres* $r_1, \ldots, r_N$ et une fonction de transition de $Q \times \{0,1\}^N$ dans $Q \times \{+1, 0, -1\}^N : \delta(q, \alpha_1, \ldots, \alpha_N) = (q', d_1, \ldots, d_N)$ telle que si $\alpha_i = 0$, alors $d_i \neq -1$.

Une *configuration* de la machine est donnée par N entiers en base 1 (les contenus de registres) et un état.

Un mouvement de la machine est une relation entre configurations : $q, k_1, \ldots, k_N \vdash_M q', k'_1, \ldots, k'_N$ ssi il existe une transition de la machine $q, \alpha_1, \ldots, \alpha_N \mapsto q', d_1, \ldots, d_N$ telle que $\alpha_i = 0$ si $k_i = 0$ et $\alpha_i = 1$ sinon et, pour tout i, $k'_i = k_i + d_i$.

Un *calcul* de la machine sur la donnée n_0 est une suite de configurations, la première étant la configuration initiale $(q_0, n_0, 0 \ldots, 0)$ et telle que la $i$ème configuration est obtenue par un mouvement de la machine sur la $i-1$ème configuration.

Une machine à registres M *accepte* l'entier n_0, si le calcul de M sur n s'arrête en **accept**.

1. Soit $k > 1$. Montrer qu'on peut construire une machine à registresM_k telle que, sur la donnée $n \in \mathbb{N}$, M_k s'arrête dans une configuration où le premier registre contient q et le deuxième registre contient r où q, r sont respectivement le quotient et le reste de la division euclidienne de n par k.
2. Soit $\Sigma \setminus \{\$, B\} = \{a_1, \ldots, a_k\}$. Pour $w \in (\Sigma \setminus \{\$, B\})^*$, on note $c_k(w)$ l'entier dont l'écriture en base $k+1$ est w.

   Montrer que, si L est récursivement énumérable, il existe une machine à 4 registres qui accepte $\{c_k(w) \mid w \in L\}$.
3. Montrer le même résultat que celui de la question précédente, mais avec une machine à 2 registres (au lieu de 4). Ind : on pourra considérer le codage de 4 entiers en un seul par $\overline{(n_1, n_2, n_3, n_4)} = 2^{n_1} \times 3^{n_2} \times 5^{n_3} \times 7^{n_4}$.
4. Montrer que le problème de savoir si une machine à 2 registres accepte au moins un entier est indécidable.